INTEGRATED COMPUTER CONTROL SYSTEM

P. J. VanArsdall
R. C. Bettenhausen
F. W. Holloway
R. A. Saroyan
J. P. Woodruff

The National Ignition Facility (NIF) design team is developing the integrated computer control system (ICCS), which is based on an object-oriented framework applicable to event-driven control systems. The framework provides an open, extendable architecture that is sufficiently abstract to construct future mission-critical control systems. Supervisory software is constructed by extending the reusable framework components for each specific application. The framework incorporates services for database persistence, system configuration, graphical user interface, status monitoring, event logging, scripting language, alert management, and access control. More than twenty collaborating software applications are derived from the common framework.

The ICCS consists of 300 front-end processors (FEPs) attached to 60,000 control points coordinated by a supervisory system. Computers running either Solaris or VxWorks operating systems are networked over a hybrid configuration of switched fast Ethernet and Asynchronous Transfer Mode (ATM). ATM carries digital motion video from sensors to operator consoles. A brief summary of performance requirements follows (Table 1).

TABLE 1. Selected ICCS performance requirements.

Requirement	Performance
Computer restart	<30 minutes
Postshot data recovery	<5 minutes
Respond to broad-view status updates	<10 seconds
Respond to alerts	<1 second
Perform automatic alignment	<1 hour
Transfer and display digital motion video	10 frames per second
Human-in-the-loop controls response	within 100 ms

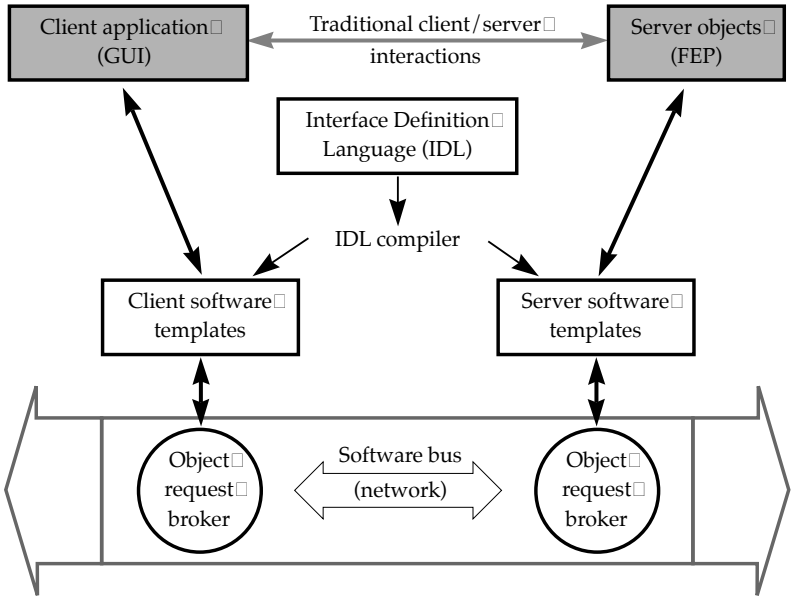
The ICCS architecture was devised to address the general problem of providing distributed control for large scientific facilities that do not require real-time supervisory controls. The framework uses the client-server software model with event-driven communications to distribute control. The framework is interoperable among different kinds of computers and transparently distributes the software objects across the network by leveraging a Common Object Request Broker Architecture (CORBA).

CORBA DISTRIBUTION

Past architectural approaches to distributed controls have relied on the technique of building large-application programming interface (API) libraries to give applications access to functions implemented throughout the architecture. This practice results in large numbers of interconnections that quickly increases the system complexity and make software modification much more difficult. To address this problem in the ICCS, software objects are distributed in a client-server architecture using CORBA.

CORBA is a standard developed by a consortium of major computer vendors to propel the dominance of distributed objects on local area networks and the World Wide Web. The best way to think of CORBA is as the universal "software bus." CORBA is a series of sophisticated, but standard sockets into which software objects can "plug and play" to interoperate with one another, even when made by different vendors. By design, CORBA objects interact across different languages, operating systems, and networks.

At a greatly simplified level, the major parts of CORBA are shown in the figure below. The interface types and methods provided by the server objects and used by the clients are defined by an industry standard Interface Definition Language (IDL). The IDL compiler examines the interface specification and generates the necessary interface code and templates into which user-specific code is added. The code in the client that makes use of CORBA objects is written as if the server was locally available and directly callable. Each computer on the network has an object request broker that determines the location of remote objects and transparently handles all communication tasks.



Software bus network implementation for CORBA distribution.
(40-00-0298-0341pb01)

Over twenty distributed software applications built from the common framework will operate the NIF control system hardware from a central control room (Figure 1). The ICCS software framework is the key to managing system complexity and, because it is fundamentally generic and extensible, it is also reusable for the construction of future projects.

Control System Architecture

The ICCS is a layered architecture consisting of FEPs coordinated by a supervisory system (Figure 2). Supervisory controls, which are hosted on UNIX workstations, provide centralized operator controls and status, data archiving, and integration services. FEP units are constructed from VME/VXI-bus or PCI-bus crates of embedded controllers with interfaces that attach to control/monitor points (e.g., motors and calorimeters). FEP software implements the distributed services needed to operate the hardware by the supervisory system. Precise triggering of fast diagnostics and controllers is handled during a two-second shot interval by the timing system, which is capable of providing triggers to 30-ps accuracy and stability. The software is distributed among the computers and provides plug-in software extensibility for attaching control/monitor points and other software services by using the CORBA protocol.

The operator console provides the human interface in the form of operator displays, data retrieval and processing, and coordination of control functions. Supervisory software is partitioned into several cohesive subsystems, each of which controls a primary NIF subsystem such as alignment or power conditioning. Several databases are incorporated to manage experimental and operations data. The subsystems are integrated to coordinate operation of laser and target area equipment.

FEPs perform sequencing, data acquisition, process control, and data reduction. The software framework includes a standard way for FEP units to be integrated into the supervisory system by providing the common distribution mechanism coupled with software patterns for hardware configuration, command, and status monitoring functions.

A segment of the control system accommodates industrial controls for which standard commercial solutions already exist. The segment is composed of a network of programmable logic controllers that reside below the FEP and attach to field devices that functionally control, for example, vacuum systems, argon gas in the beam tubes, and thermal gas conditioning for amplifier cooling. This segment also observes the independent safety interlock system, which monitors doors, hatches, shutters, and other sensors to establish and display the hazard levels in the facility. Potentially hazardous equipment is permitted to operate only when conditions are safe. Interlocks

FIGURE 1. Photograph of the NIF computer control room. (40-00-1298-2628pb01)

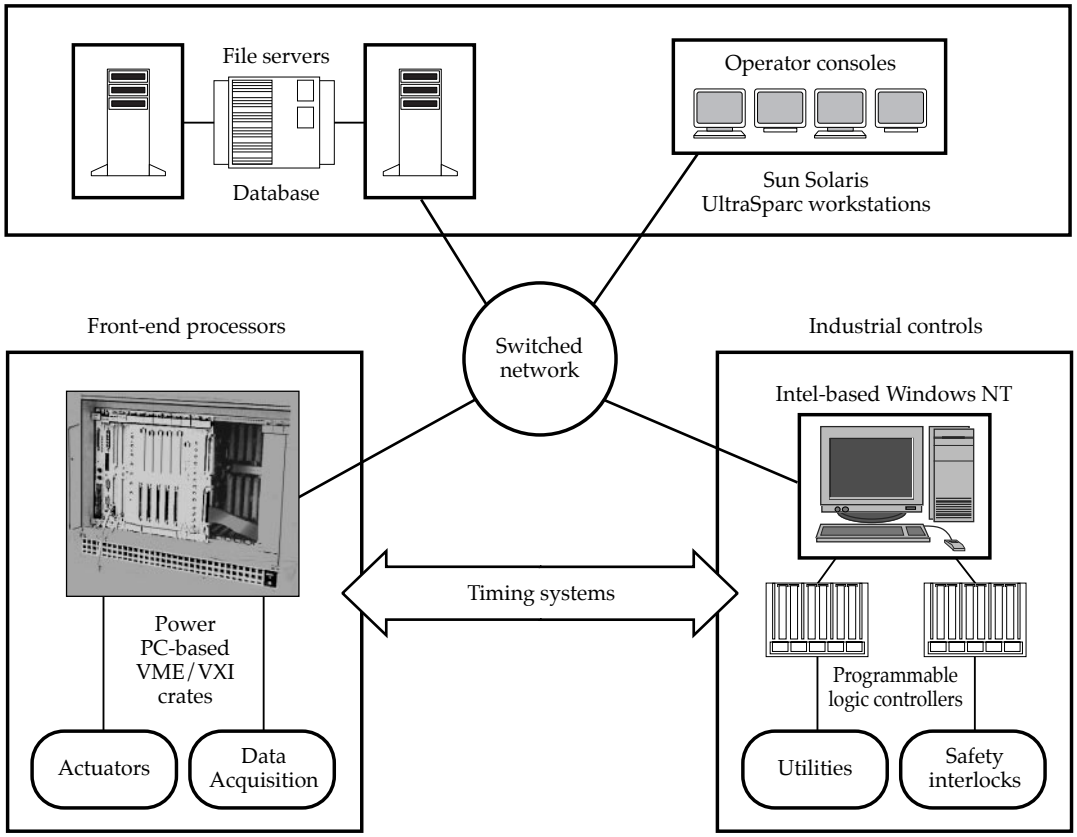


FIGURE 2. Integrated computer control system architecture. (40-00-1298-2627pb01)

function autonomously to ensure safety without dependency on the rest of the control system.

There are eight supervisory software applications that conduct NIF shots in collaboration with 19 kinds of FEPs as shown in Figure 3. The ICCS is partitioned into several loosely coupled systems that are easier to design, construct, operate, and maintain. Each subsystem is composed of a supervisor and associated FEPs. The eighth supervisor is the shot director, which is responsible for conducting the shot plan, distributing the countdown clock, and coordinating the other seven supervisory applications.

Seven subsystems comprise the primary supervisory controls. The Alignment Supervisor provides coordination and supervision of laser wavefront control and laser component manual and automatic alignment. The Laser Diagnostics Supervisor provides functions for diagnosing performance of the laser by collecting integrated, transient, and image information from sensors positioned in the beams. The Optical Pulse Generation Supervisor provides temporally and spatially formatted optical pulses with the correct energetics and optical characteristics required for each of the beams. The Target Diagnostics Supervisor coordinates the collection of data from a diverse and changing set of instruments. The Power-Conditioning Supervisor is responsible for high-level control and management of high-voltage power supplies that fire the main laser amplifiers. The Pockels Cell Supervisor manages operation of the plasma electrode Pockels cell optical switch that facilitates multipass amplification within the main laser amplifiers. The Shot Services Supervisor provides monitoring of environmental and safety parameters as well as control of programmable logic controller subsystems.

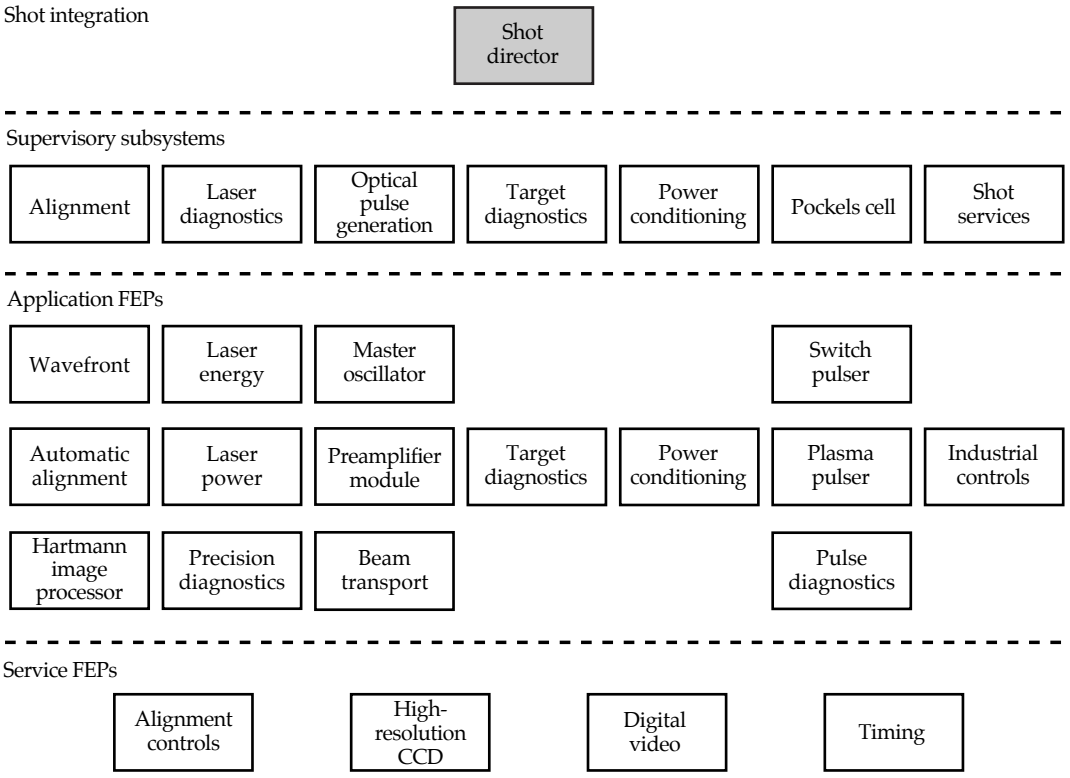


FIGURE 3. Software applications in the NIF control system. (40-00-0298-0324pb02)

Computer System and Network

Figure 4 shows the NIF computer and network architecture, which is composed of 30 UNIX workstations, 300 FEPs, and several hundred more embedded controllers (not shown). The main control room contains seven graphics consoles, each of which houses two workstations with dual displays. The supervisors are normally operated from a primary console, although the software can easily be operated from adjacent consoles or remote graphics terminals located near the front-end equipment. File servers provide archival databases as well as centralized management services necessary for coordinating the facility operation.

The network design utilizes both Ethernet and ATM technologies to take advantage of the best features of each. ATM provides time-sensitive video transport, whereas Ethernet provides connectivity for the large majority of systems. The design utilizes 155 Mbit/s ATM and Ethernet at both 10 and 100 Mbit/s speeds, depending on expected traffic requirements. The operator consoles and file servers have both ATM and 100 Mbit/s Ethernet connections, while most FEPs have either 10 or 100 Mbit/s connections made through Ethernet switches. Given the low cost and performance advantages of Ethernet switches relative to shared Ethernet hubs, switches with 100 Mbit/s uplinks will be used.

TCP/IP (Transmission Control Protocol/Internet Protocol) is the protocol used for reliable data transport between systems, either over Ethernet or ATM. TCP provides retransmission of packets in the event that one is lost or received in error. The only traffic not using TCP will be digitized video and network triggers. Video is transferred using the ATM adaptation layer 5 (AAL5) protocol. Network triggers are broadcast to many end-nodes simultaneously using multicast protocols.

The network supports the transport of digitized motion video in addition to the more typical control, status, and shot data.

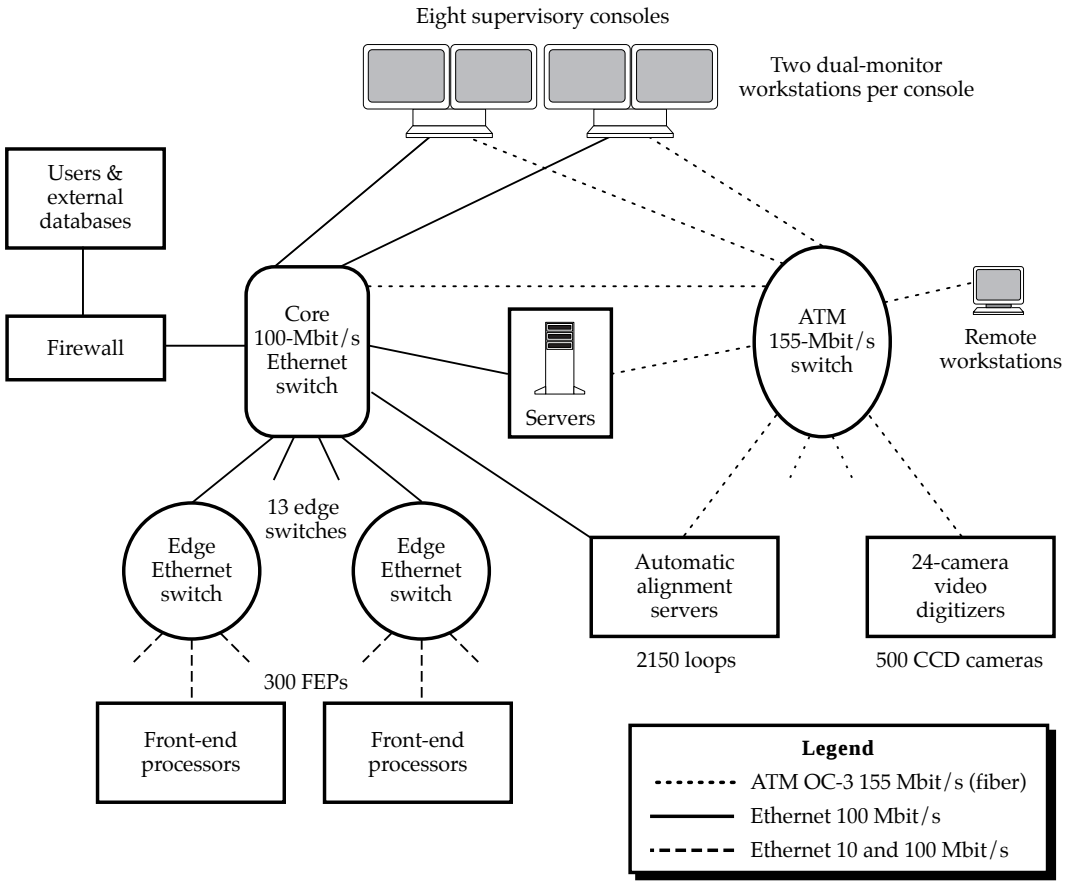


FIGURE 4. NIF computer system and network architecture.
(40-00-1298-2629pb01)

The network transports video streams at 10 frames per second (about 25 Mbit/s) between special FEPs that digitize camera images and operator workstations that can display at least two concurrent streams. Video compression is not used because of the high cost of encoding the video stream.

Digitized video is sent via the ATM application program interface (API) using the ATM quality of service capabilities. The API provides an efficient method of moving large, time-sensitive data streams, resulting in higher frames/second rates with lower central processing unit (CPU) utilization than alternative approaches, which is an important consideration for the video FEPs and console workstations. Performance testing of the prototype video distribution system indicates that 55% of the FEP CPU (300-MHz UltraSparc AXI) is used to broadcast three streams, while 10% of the operator workstation CPU (300-MHz UltraSparc 3D Creator) is utilized for each playback stream.

Estimates of the peak traffic requirements for the various subsystems were analyzed as a basis for the network design. The expected peak traffic flows between subsystems in terms of messages per second and message size were specified. This data was combined and analyzed to determine peak throughputs into and out of each network-attached device.

A discrete event simulation was created to evaluate the performance of key network scenarios. For example, it was shown that "network triggers" in the millisecond regime could be reliably broadcast between computers in the network via the user datagram protocol. One application using network triggers is the arming of the video FEPs to enable digitizing the laser pulse during shots. The network trigger tactic will save $250K over a hardware implementation.

If bandwidth requirements increase in the future, the network architecture allows

the integration of Gbit/s Ethernet and 622 Mbit/s ATM technologies in a relatively straightforward manner.

Software Framework

The ICCS supervisory software framework is a collection of collaborating abstractions that are used to construct the application software. Frameworks[1] reduce the amount of coding necessary by providing prebuilt components that can be extended to accommodate specific additional requirements. The framework also promotes code reuse by providing a standard model and interconnecting CORBA backplane that is shared from one application to the next.

Components in the ICCS framework are deployed onto the file servers, workstations, and FEPs, as shown generically in Figure 5. Engineers specialize the framework for each application to handle different kinds of control points, controllers, user interfaces, and functionality. The framework concept enables the cost-effective construction of the NIF software and provides the basis for long-term maintainability and upgrades.

The following discussion introduces the framework components that form the basis of the ICCS software.

Configuration—a hierarchical organization for the static data that define the hardware control points accessible to the ICCS. Configuration provides a taxonomic system

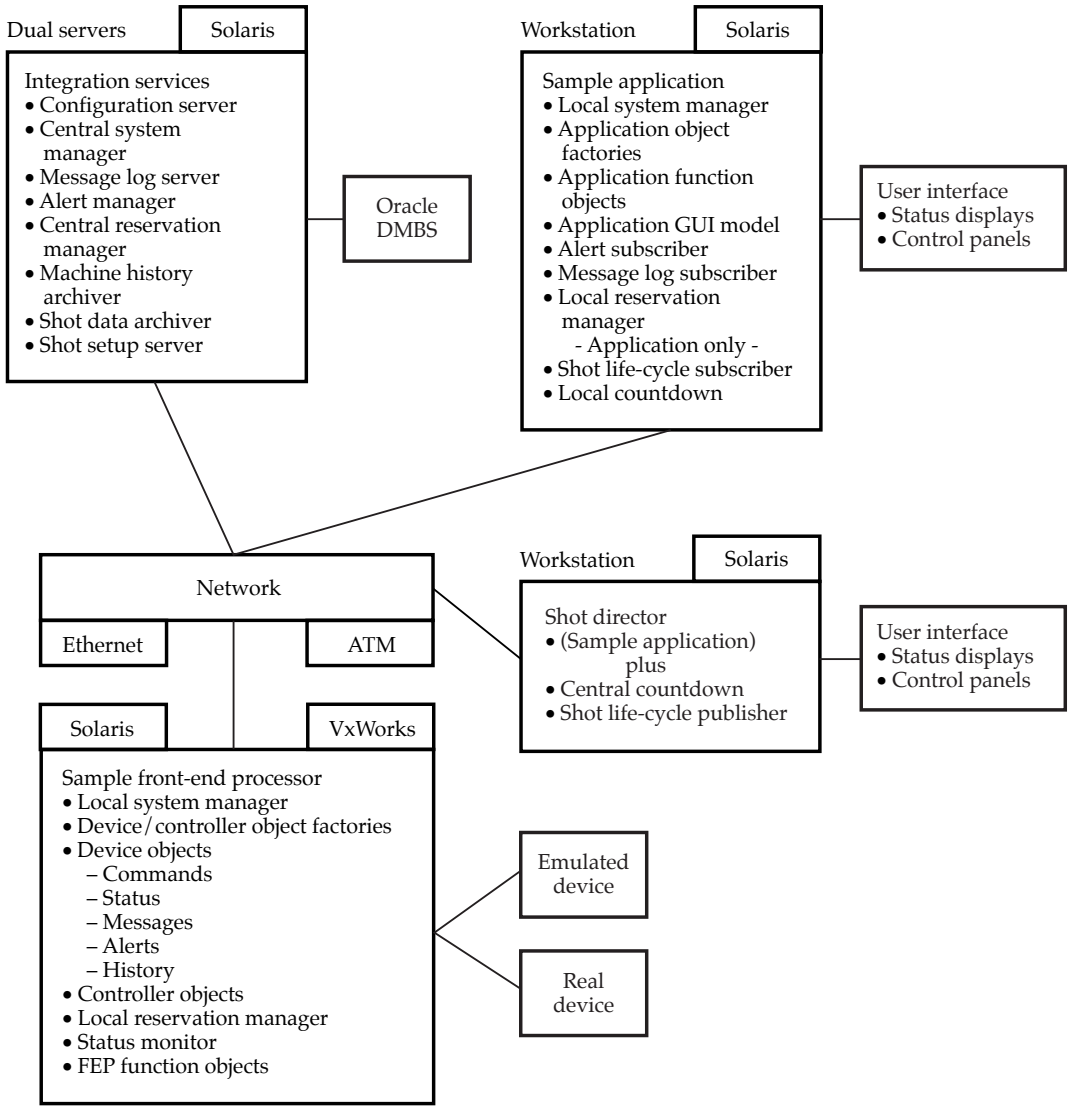


FIGURE 5. Deployment of ICCS framework objects into a sample application and FEP on networked computers. (40-00-1298-2630pb01)

used as the key by which clients locate devices (and other software services) on the CORBA bus. During normal operation, configuration provides to clients the CORBA references to all distributed objects. An important responsibility of configuration is the initialization of FEPs during start-up. Configuration data are stored in the database and describe how and where the control hardware is installed in the system. Calibration data for sensors, setpoints for alignment devices, and I/O channels used by devices on interface boards are examples of static data managed by configuration. During ICCS start-up, this framework collaborates with an object factory located in the FEP. Using the data and methods stored in the configuration database, the object factory instantiates, initializes, and determines the CORBA reference for each device and controller object in the FEP.

Status Monitor—a device that provides generalized services for broad-view operator display of device status information using the publisher–subscriber model of event notification. The status monitor operates within the FEP observing devices and notifies other parts of the system when the status changes by a significant amount. Network messages are only generated when changes of interest occur.

Sequence Control Language—a language used to create custom scripting languages for the NIF applications. The service automates sequences of commands executed on the distributed control points or other software artifacts. After full implementation of the design, operators will create and edit sequences by selecting icons that represent control constructs, Boolean functions, and user-supplied methods from a visual programming palette. The icons are then interconnected to program the sequence, and any Boolean conditions or method arguments needed are defined to complete the sequence script.

Graphical User Interface (GUI)—the method through which all human interaction with the ICCS will take place and which will be displayed on control room consoles or other workstations in the facility. The GUI is implemented as a framework to ensure consistency across the applications. Commercial GUI development tools are used to construct the display graphics. This framework consists of guidelines for look and feel as well as many common graphical elements.

Message Log Server—a device that provides event notification and archiving services to all subsystems or clients within the ICCS. A central server collects incoming messages and associated attributes from processes on the network, writes them to appropriate persistent stores, and also forwards copies to interested observers such as GUI windows.

Alert System Manager—a component that raises an alert for any application encountering a situation that requires immediate attention, which then requires interaction with an operator for the control system to proceed. The alert system records its transactions so that the data can be analyzed after the fact.

Reservation Manager—a system that manages access to devices by giving one client exclusive rights to control or otherwise alter a control/monitor point. The framework uses a lock-and-key model. Reserved devices that are "locked" can only be manipulated if and when a client presents the "key."

System Manager—a component that provides services essential for the integrated management of the ICCS computer network. This component ensures that necessary processes and computers are operating and communicating properly. Services include orderly system start-up, shutdown, and watchdog process monitoring.

Machine History Archiver—a system that gathers information for analysis about NIF operational performance to improve efficiency and reliability. Examples of such information are component adjustments, abnormal conditions, operating service, periodic readings of sensors, and reference images.

Generic FEP—a component that pulls together the distributed aspects of the other frameworks (in particular the system manager, configuration, status monitor, and reservation frameworks) by adding unique classes for supporting device and controller interfacing. These classes are

responsible for hooking in CORBA distribution as well as implementing the creation, initialization, and connection of device and I/O controller objects. The generic FEP also defines a common hardware basis including the processor architecture, I/O board inventory, device drivers, and field-bus support. The FEP application developer extends the base software classes to incorporate specific functionality and control logic.

Shot Data Archiver—a server working in collaboration with the System Manager to assure that requested shot data are delivered to a disk staging area. The ICCS is responsible for collecting the data from diagnostics, making the data immediately available for "quick look" analysis, and delivering the data to an archive.

Shot Setup Server—a server working in collaboration with the ICCS shot director to manage shot setup plans. These plans contain the experimenter's goals for a shot and the hardware setup derived from the goals.

Software Development Environment

The ICCS incorporates Ada95, CORBA, and object-oriented techniques to enhance the openness of the architecture and long-term maintainability of the software. C++ is also supported for the production of graphical user interfaces and the integration of commercial software. Software development of an expected 500,000 lines of code is managed under an integrated software engineering process (Figure 6) that covers the entire life cycle of design, implementation, and maintenance. The object-oriented design is captured in the Rose design tool in UML (unified modeling language) notation that maintains schematic drawings of the software architecture.

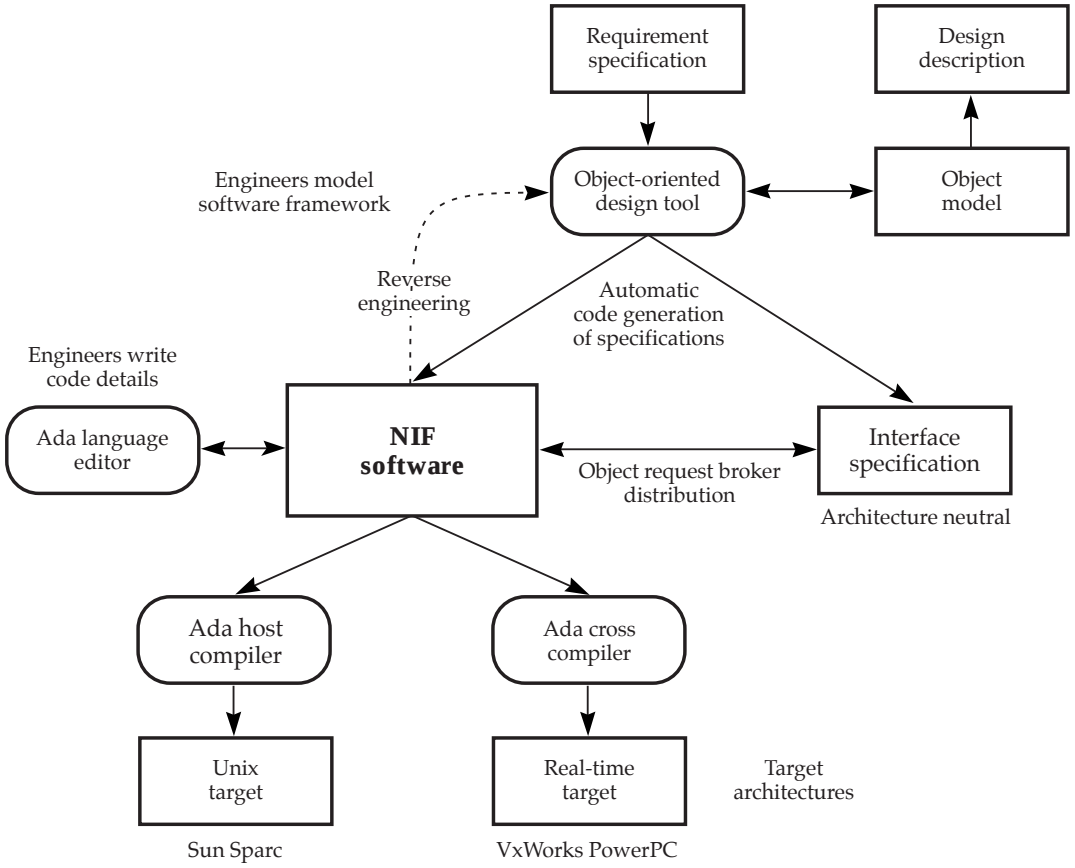


FIGURE 6. Flowchart of the ICCS software engineering process incorporating model-driven design techniques. (40-00-1298-2631pb01)

Developers analyze detailed requirements and express them as scenarios that help determine how the software will be organized. In essence, the Rose tool is used to model the public interfaces and interactions between major software entities (i.e., the classes). Rose automatically generates Ada code specifications corresponding to the class interface and type definitions. The developer fills in the detailed coding necessary to implement the private contents of each class. Rose generates IDL for classes that are distributed, which is passed through the IDL compiler to generate Ada or C++ skeleton code as before. The design description is a narrative document that explains the object-oriented model and contains other information necessary for implementation and maintenance.

Source code is compiled for a variety of target processors and operating systems. Current development is either self-hosted to Solaris on Sparc processors or cross-compiled for VxWorks on PowerPC processors. The models, sources, binaries, and run-time images are version-controlled by the Apex configuration management system, which allows the frameworks and applications to be independently developed by different engineers, each having a protected view of the other components.

Summary

The ICCS is being developed using the iterative approach to software construction[2] that is proven effective for projects whose requirements continue to evolve. Five iterations in the "Light" series (Figure 7) are planned leading to facility deployment in 2000 when the first 8 of the 192 beams will be operated. Each new release will follow an updated plan addressing the greatest risks to the architecture while increasing the functionality delivered to the Project.

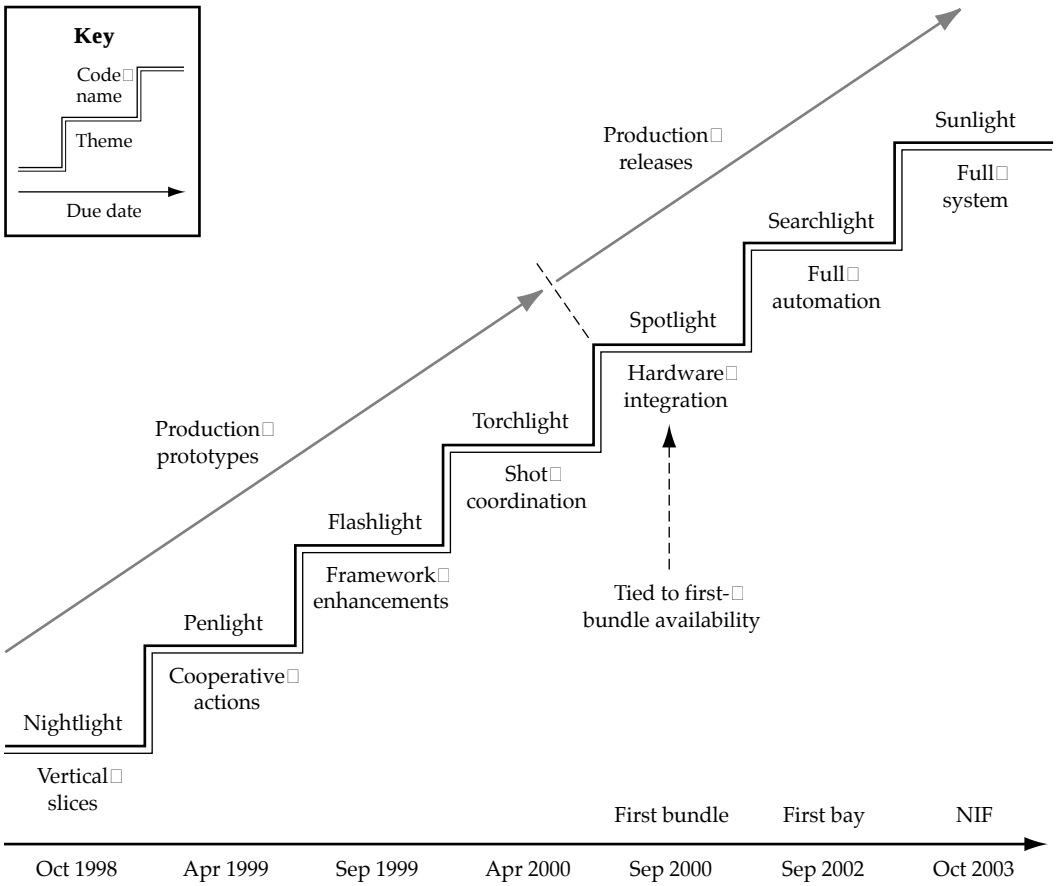


FIGURE 7. Light series of NIF software construction. (40-00-1298-2632pb01)

The first release, "Nightlight," was completed this fall and submitted to independent test. Nightlight delivered 120,000 source lines of tested code—about 20% of the anticipated total—which exercised the framework in initial vertical slices of all supervisory and FEP applications. Nightlight also established an independent test process and the procedures by which quality software could be assured.

Construction of the ICCS incorporates many of the latest advances in distributed computer and object-oriented software technology. Primary design goals are to provide an extensible and robust architecture that can be maintained and upgraded for decades. Software engineering is a managed process utilizing model-driven design to enhance the product quality and minimize future maintenance costs. The ICCS framework approach permits software reuse and allows the system to be constructed within budget. As an added benefit, the framework is sufficiently abstract to allow future control systems to take advantage of this work.

Acknowledgments

The authors wish to acknowledge the contributions of the following colleagues without whose efforts this work would not be possible: G. Armstrong, R. Bryant, R. Carey, R. Claybourne, T. Dahlgren, F. Deadrick, R. Demaret, C. Estes, K. Fong, M. Gorvad, C. Karlsen, B. Kettering, R. Kyker, L. Lagin, G. Larkin, G. Michalak, P. McKay (Sandia National Laboratories, Albuquerque, NM), M. Miller, V. Miller Kamm, C. Reynolds, R. Reed, W. Schaefer, J. Spann, E. Stout, W. Tapley, L. Van Atta, and S. West.

Notes and References

1. V. Swaminathan and J. Storey, *Object Magazine*, April 1988, pp. 53–57.
2. B. Boehm, *IEEE Computer*, May 1988, pp. 61–72.